METHOD FOR SOFT-PROGRAMMING AN ELECTRICALLY ERASABLE NONVOLATILE MEMORY DEVICE, AND AN ELECTRICALLY ERASABLE NONVOLATILE MEMORY DEVICE IMPLEMENTING THE SOFT-PROGRAMMING METHOD

5   BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a method for soft-programming an electrically erasable nonvolatile memory device and to an electrically erasable nonvolatile memory device implementing the soft-programming method.

10   Description of the Related Art

As is known, nonvolatile memories comprise a memory array formed by memory cells arranged in rows and columns, in which wordlines connect gate terminals of memory cells arranged on the same row and bitlines connect drain terminals of memory cells arranged on the same column.

15   It is likewise known that in a nonvolatile floating-gate memory cell the storage of a logic state is carried out by programming the threshold voltage of the memory cell through the definition of the quantity of electrical charge stored in the floating-gate region.

According to the information stored, the memory cells are

20   distinguished into erased memory cells (logic state stored "1"), in which no electrical charge is stored in the floating-gate region, and in written or programmed memory cells (logic state stored "0"), in which there is stored in the floating-gate region an electrical charge sufficient to determine a sensible increase in the threshold voltage of the memory cells.

25   In nonvolatile memories, moreover, the memory array is generally divided into sectors, each one of which is formed by a set of memory cells on

1

which it is possible to carry out simultaneously the same operation, generally an erasure operation. In particular, in nonvolatile memories organized into sectors it is possible to carry out reading and programming of individual memory cells of a sector and simultaneous erasure of all the memory cells of the sector, and the

5   latter operation is rendered possible by the fact that the memory cells belonging to the same sector have source terminals connected to one another.

Erasure of a nonvolatile memory carried out sector by sector is a cumulative operation, *i.e.*, it acts simultaneously and indistinctly on all the memory cells of the sector, and is very complex in so far as it requires not just some

10   preparatory steps to be carried out prior to executing the erasure proper, during which the electrical charges in the floating-gate region are extracted and consequently the threshold voltage of the memory cells themselves is reduced, but also requires verifications and possible modifications subsequent to the erasure proper in the case where the result of the erasure is not fully satisfactory.

15   In particular, to erase a sector there is first of all performed an operation of pre-conditioning, also known as "program-all-0", *i.e.*, an operation whereby all the memory cells of the sector are taken to the programmed state irrespective of their current state. This is due to the fact that, if there were to be erased a sector in which some of the memory cells are written but others are

20   already erased, during the erasure there would occur an over-erasure of the memory cells already erased, which, with a fair degree of likelihood, would become depleted memory cells, *i.e.*, memory cells that have negative threshold voltage and thus drain a current even when their gate terminal is set at ground voltage. These depleted memory cells are particularly troublesome in so far as they simulate the

25   constant presence of erased memory cells in the respective columns to which they belong and thus cause all the memory cells belonging to said columns to be read as erased irrespective of their actual state.

In order to prevent this phenomenon and to render the history of all the memory cells belonging to the same sector uniform, there is hence performed

2

the step of writing the entire sector, and then, following upon the operation of pre-conditioning, all the memory cells of the sector are programmed.

Next, there is carried out the erasure step proper, during which the electrical charges in the floating-gate region are extracted, and consequently the

5   threshold voltage of the memory cells is reduced.

In order to carry this out, between the source terminal and the gate terminal of each memory cell there is applied a high electric field such as to enable the electrons to abandon the floating-gate region thanks to the so-called Fowler-Nordheim tunnelling effect.

10   The application of the electrical potential necessary to extract the electrical charges from the floating-gate region can occur in different ways. One of the methodologies that can, for example, be used to extract the electrical charges from the floating-gate region is known in the literature as "negative gate erasure" and basically envisages leaving floating the drain terminal of the memory cell to be

15   erased and applying to the gate terminal a negative voltage pulse having typically an amplitude of 10 V and a time duration of the order of 10 ms, and to the source and body terminals a succession of voltage pulses the amplitude of which is variable in staircase fashion from a minimum value of 3 V to a maximum value of 8 V, with steps of 300 mV.

20   At the end of the erasure pulse, there is then carried out an operation of verification on all the memory cells of the sector in order to check the value of their threshold voltages, and this verification is made by performing a marginated reading that will guarantee correct recognition of the memory cell in the normal reading mode.

25   In particular, the verification operation scans all the memory cells of the sector and is interrupted when a memory cell that does not pass the test is found. At this point the next erasure pulse is applied. Hence, the erasure process consists in the application of an erasure pulse, followed by a subsequent step of verification until all the memory cells have a threshold voltage lower than a

reference threshold voltage, which is the threshold voltage of the reference memory cell used during the verification operation. Once the voltage pulses applied to the source and body terminals have reached their maximum amplitude, if the memory cells of the sector are not yet all erased, then there is envisaged a

5   second step of pure electrical erasure, during which further erasure pulses are applied both to the gate terminals and to the source and body terminals of the memory cells of the sector, up to a given maximum number, in which the amplitude of the pulses applied to the source and body terminals of the memory cells is equal to the maximum amplitude envisaged.

10   At the end of pure electrical erasure, all the memory cells of the sector have threshold voltages presenting a basically gaussian distribution, on which there is superimposed a possible "tail" due to the depleted memory cells.

The sector erasure step cannot, however, yet be deemed concluded because it is still necessary to ensure that there will not be depleted memory cells

15   that can induce errors during the reading step. As previously mentioned, in fact, since these memory cells have a negative threshold voltage and thus drain a current even when their gate terminals are set at ground voltage, they can vitiate the subsequent operation of reading of the memory device in so far as they simulate the constant presence of erased memory cells in the respective columns

20   to which they belong and thus cause all the memory cells belonging to these columns to be read as erased irrespective of their actual state.

Hence, the erasure step proper is followed by a step of search and re-programming of the depleted memory cells, known also as "soft-programming", in which there is verified the presence of a leakage current in the columns of the

25   memory array by biasing all the rows of the array at ground voltage.

When a column presenting this anomaly is identified, then the first memory cell of the column is addressed, and a programming pulse having a pre-set amplitude is applied to its gate terminal in order to shift slightly the threshold voltage of the memory cell towards higher values.

4

This is followed by reading of the second memory cell of the same column: if no leakage current is present in the memory cell, then this means that the depleted memory cell was the preceding one that has already been recovered; otherwise, the memory cell considered is programmed, and so on up to the end of the column.

Once the end of the column has been reached, the verification operation is then repeated, and in the case where a leakage current is still present, the procedure described above is repeated, increasing, however, the amplitude of the programming pulse applied to the gate terminals of the memory cells during programming.

In practice, soft-programming is never carried out on a single memory cell at a time, but rather on a given number of memory cells at a time that is linked to the number of programming circuits (program loads) with which the memory device is provided, currently sixteen. In particular, soft-programming is carried out by addressing a number of memory cells equal to the number of program loads and by soft-programming only those memory cells addressed that effectively require this operation. The number of memory cells that are simultaneously soft-programmed is generally known, in the technical field, as "soft-programming parallelism" or "soft-programming multiplicity".

To compete in the sectors of emerging markets of portable electronic apparatuses, such as, for example, MP3 players and digital cameras, nonvolatile memory devices are currently subject to a revision of their traditional specifications; in particular they are the subject of in-depth studies aimed, among other things, at reducing erasure time.

A contribution to reduction in erasure time could come from the reduction of soft-programming time, which, at least in principle, could be obtained by increasing the parallelism with which this operation is performed.

For example, reduction in soft-programming time could be obtained simply by doubling the number of program loads, but this would entail, in the case

5

where all the memory cells addressed were to require soft-programming, a doubling of the current absorbed in the soft-programming step.

In order to generate this current, it is possible to act in two ways: either to generate the current necessary outside of the memory device, using an appropriate current generator, and supply it to a purposely provided pin of the memory device, or else to double the size of the charge pump, which, in the memory device, is dedicated to the generation of high currents.

The first solution runs, however, counter to the trend of current markets of semiconductor memory devices, which, instead, requires reduction of both the number of voltage supplies in the memory device and their absolute voltage value, whilst the second solution would lead to a significant increase in the area occupied by the charge pump on the silicon and thus to an increase in the cost of the memory device.

BRIEF SUMMARY OF THE INVENTION

A purpose of the present invention is to provide a method for erasing a nonvolatile memory device that will enable a reduction in erasure time with respect to nonvolatile memory devices according to the prior art.

A further purpose of the present invention is to provide a nonvolatile memory device having erasure time shorter than that of nonvolatile memory devices according to the prior art.

According to the present invention, a method for soft-programming an electrically erasable nonvolatile memory device is provided.

According to the present invention an electrically erasable nonvolatile memory device is also provided.

6

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the present invention there is now described a preferred embodiment, purely by way of non-limiting example thereof with reference to the attached drawings, in which:

Figure 1 shows a nonvolatile memory device architecture; and

Figures 2, 3 and 4 show flowcharts of the operations relative to the erasure method according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

By way of example, Figure 1 is a schematic illustration of the architecture of a memory device of the type described above, in which the reference number 1 designates, as a whole, the nonvolatile memory device, the reference number 2 designates the memory array, number 3 designates the sectors of the memory array 2, number 4 designates the memory cells, number 5 designates the wordlines, number 6 designates the bitlines, number 7 designates the common line to which the source terminals of all the memory cells 4 belonging to the same sector 3 are connected, number 8 designates the row decoder, and finally number 9 designates the column decoder.

The present invention draws inspiration from the result of a study carried out by the present Applicant on the statistical distribution of depleted memory cells in nonvolatile memory devices. In particular, this study has highlighted that the number of depleted memory cells in a nonvolatile memory device is on average equal to 5% of the total number of memory cells of the memory array, that the depleted memory cells are generally not uniformly distributed within the memory array but rather are localized in a given area of the memory array, and that this different behaviour (depletion) of some memory cells of a sector of the memory array with respect to the remaining part of the memory cells of the sector is explicable with a defectiveness localized in a given area of silicon on which the memory array is made.

Thanks to this and to the fact that typically the bits of a word are not stored in adjacent memory cells of a sector but rather are stored in memory cells distributed in different areas of the sector according to a definite criterion (for example, spaced by 128 columns from one an other, for a device made up of 4096

5   columns), statistically the probability that a large number of memory cells which store the bits of the same word will be depleted is extremely reduced, so that typically there will be reading errors in just some of the bits of a word stored.

In other words, the distribution of the storage of the bit of the same word within a sector of the memory array, combined with the fact that the

10   defectiveness of the silicon on which the memory array is made is typically localized, renders extremely unlikely that a large number of memory cells that store the bits of the same word will fall within the same defective silicon area, so that typically only a few bits of the word will be affected by the depletion problem.

Drawing inspiration from the experimental findings highlighted above,

15   the idea underlying the present invention is simply that of soft-programming and verifying the memory cells with a multiplicity that is twice (for example 32 bits) that of the one used in programming (for example 16 bits), without any need of supplying from outside the larger current necessary or of increasing the dimensions of the charge pump inside the memory device.

20   This can be obtained in a simple manner by appropriately modifying the soft-programming algorithm in such a way that it will operate both with the multiplicity used in programming and with a double multiplicity, *i.e.*, twice that used in programming. In particular, in normal operation the soft-programming algorithm operates at double multiplicity (in the example considered, 32 bits), *i.e.*, at the

25   maximum speed, whilst in the case where there is an excessive number of depleted memory cells and it is not possible to soft-program simultaneously the number of memory cells envisaged for double-multiplicity operation because the maximum current required exceeds the capacity of the charge pump, then the soft-programming algorithm passes to operating at single multiplicity, *i.e.*, with the

multiplicity used in programming (in the example considered, 16 bits), during which the charge pump is able to supply the maximum current required.

After that two sets of memory cells (in the example considered, two sets of 16 memory cells each) have been programmed at single multiplicity, the
5 soft-programming algorithm returns to operating at double multiplicity.

It can readily be shown that, if w is the number of bits of a word, b is the number of bits stored in depleted memory cells, hereinafter for reasons of brevity referred to as depleted bits, and p is the probability density function of finding b depleted bits on w bits, with $p = 100*Fb(b)$ $0 \le p \le 100$ and $0 \le b \le w$, if
10 the aim is for the soft-programming algorithm to pass to multiplicities lower than the maximum one for $a\%$ (e.g., $0 < a < 100$) of the times, for one programming we must have $p > a$, i.e.,

$$100 \cdot Fb(p) > a \Rightarrow b = Fb^{-1}\left(\frac{a}{100}\right)$$

where b is the maximum number of depleted bits per word that is found $a\%$ of the
15 times.

If B is the maximum number of bits that it is possible to soft-program at a time, which, in the case in point, coincides with w, the maximum multiplicity that can be reached is:

$$N = INT\left(\frac{B}{b}\right) = INT\left(\frac{B}{Fb^{-1}\left(\frac{a}{100}\right)}\right) \cong INT\left(\frac{w}{Fb^{-1}\left(\frac{a}{100}\right)}\right)$$

20 Considering, for example, memory devices manufactured by using a flash fabrication process known by the code t7x, where the depleted memory cells are found to be 4.3% of the total, from a statistical analysis carried out on typical memory devices the following probability-density function was obtained:

$P(b \le 0) = 50.78$	$P(b \le 9) = 0.000$
$P(b \le 1) = 17.96$	$P(b \le 10) = 0.00$

$$P(b \le 2) = 3.125 \qquad P(b \le 11) = 0.00$$

$$P(b \le 3) = 0.781 \qquad P(b \le 12) = 0.00$$

$$P(b \le 4) = 0.000 \qquad P(b \le 13) = 0.00$$

$$P(b \le 5) = 0.000 \qquad P(b \le 14) = 0.00$$

5

$$P(b \le 6) = 0.000 \qquad P(b \le 15) = 0.00$$

$$P(b \le 7) = 0.000 \qquad P(b \le 16) = 0.00$$

$$P(b \le 8) = 0.000$$

Hence, if we wish the probability of passing to lower multiplicity to be 4%, then we have $b = 2$, and for $w = 16$ we obtain $N = 4$, for a total of

10    $N * w = 4 * 16 = 64$ soft-programmable bits at a time.

If, instead, we use $N = 2$, for $w = 16$ ($N * w = 2 * 16 = 32$ soft-programmable bits at a time), we have $b = 4$ with a probability of passing to multiplicity $N = I$ ($N * w = I * 16 = 16$ soft-programmable bits at a time) approximately zero.

15   We shall now see specifically, with reference to the flowcharts of Figures 2, 3 and 4, how the soft-programming algorithm must be modified for implementing the idea underlying the present invention.

In particular, Figure 2 presents the flowchart of the operations corresponding to the main routine of the soft-programming algorithm according to

20   the present invention and indicated in Figure 2 by "Soft-Program", whilst Figures 3 and 4 illustrate two routines called by the main routine "Soft-Program" and indicated, respectively, by "SP_Pulse" and "L_SPX16".

Furthermore, for reasons of simplicity of description, there will be considered the case where soft-programming is carried out with 16-bit single

25   multiplicity and 32-bit double multiplicity. It will, however, be immediately clear how what has been said concerning the example considered in Figures 2, 3 and 4, is applicable just as it stands to any single and double soft-programming multiplicity that may be adopted.

10

With reference to Figure 2, the main routine "Soft-Program" envisages that there is initially set (shown as SET SPX32 in block 100) a 32-bit double soft-programming multiplicity and that the soft-programming multiplicity in use is stored in an appropriate variable SPX16_MEM (block 100). In particular,

5   the variable SPX16_MEM is a variable used for modifying the flow of operations according to the soft-programming multiplicity set and is set at a first logic value, for example "1", if 16-bit single soft-programming multiplicity is set and at a second logic value, in the example considered "0", if 32-bit double soft-programming multiplicity is set.

10   Next, a verification pulse is applied to a number of memory cells equal to the soft-programming multiplicity set (block 110), *i.e.*, 32 memory cells if the variable SPX16_MEM assumes the logic value "0", or else 16 memory cells if the variable SPX16_MEM assumes the logic value "1", and it is then verified whether there are depleted memory cells or whether, instead, all the memory cells

15   are erased (block 120). For example, the outcome of this verification can conveniently be stored in a logic-state indicator (flag), designated in the figure by "DATA_OK" at block 120, which is set at a first logic value, for example "0", if there are depleted memory cells, and at a second logic state, in the example considered "1", if all the memory cells are erased.

20   If there are depleted memory cells (output NO from block 120), then the routine SP_Pulse is run, the flowchart of which is illustrated in Figure 2, whereby soft-programming of the memory cells with single or double soft-programming multiplicity is carried out (block 130), whereas if all the memory cells are erased (output YES from block 120), then the next row is addressed (shown as

25   INCREMENT ROW in block 140), and it is then verified whether the last row address has been reached (block 150). If the last row address has not been reached (output NO from block 150), then control returns to block 110, whereas if the last row address has been reached (output YES from block 150), then the next column is addressed (shown as INCREMENT COLUMN in block 160), and it is

11

verified whether the last column address has been reached (block 170). If the last column address has not been reached (output NO from block 170), then control returns to block 110, whereas if the last column address has been reached (output YES from block 170), then the main routine "Soft-Program" terminates. Figure 3

5   illustrates the flowchart of the operations corresponding to the routine SP_Pulse.

According to what is illustrated in Figure 3, the routine SP_Pulse initially envisages that a number of memory cells equal to the soft-programming multiplicity set will be soft-programmed and stored in the variable SPX16_MEM (block 200).

10   Next, the content of the variable SPX16_MEM, which stores the soft-programming multiplicity set, is verified (block 210).

If 16-bit single soft-programming multiplicity is set (that is, SPX16_MEM = 1? and output YES from block 210), then the routine L_SPX16 is run (block 220), the flowchart of which is again illustrated in Figure 4, whereas if

15   32-bit double soft-programming multiplicity is set (output NO from block 210), then a verification pulse is applied to a number of memory cells equal to the soft-programming multiplicity set (block 230), *i.e.*, 32 memory cells, and it is then verified whether amongst them there are depleted memory cells or whether instead they are all erased (whether DATA_OK? = 0 or 1, respectively, at block

20   240).

If all the memory cells to which the verification pulse has been applied are erased (output 1 or YES from block 240), then 32-bit double soft-programming multiplicity is set (block 250), and control returns to block 140, indicated by block A in Figures 2 and 3, of the main routine "Soft-Program" (Figure

25   2, above), whereas if amongst the memory cells to which the verification pulse has been applied there are depleted memory cells (output 0 or NO from block 240), then it is verified whether a pre-set maximum number of attempts of soft-programming has been reached (block 260).

12

If this maximum number of attempts has not yet been reached (MAX ATTEMPT? output is NO from block 260), then control returns to block 200, whereas if the maximum number of attempts has been reached (MAX ATTEMPT? output is YES from block 260), then 16-bit single soft-programming multiplicity is

5   forced, and this is stored in the variable SPX16_MEM (shown as SET SPX16 in block 270).

Next, a verification pulse is applied to a number of memory cells equal to the current soft-programming multiplicity (16-bit) (block 280) and it is then verified whether amongst these there are depleted memory cells or whether,

10   instead, they are all erased (whether DATA_OK? = 0 or 1, respectively, at block 290).

If amongst the memory cells to which the verification pulse has been applied there are depleted memory cells (output is 0 or NO from block 290), then there is called again the routine SP_Pulse, whereas if all the memory cells to

15   which the verification pulse has been applied are erased (output is 1 or YES from block 290), then the next set of 16 memory cells is selected (shown as INCREMENT SENSE in block 300), and it is then verified whether the last row or column address has been reached (block 310). If the last row or column address has not been reached (output NO from block 310), then the operations resume

20   from the ones described previously with reference to block 290, whereas if the last row or column address has been reached (output YES from block 310), then the forcing of 16-bit single soft-programming multiplicity is removed ("cleared", as shown as CLEAR SPX16 in block 320), and the storage of this multiplicity is carried out in block 270 (block 320).

25   Next, control returns to block 250 previously described, in which there is again set 32-bit double soft-programming multiplicity (shown as SET SPX32 in block 250), and thence block 140 of the main routine "Soft-Program" is reached.

Figure 4 illustrates the flowchart of the operations corresponding to the routine L_SPX16.

According to what is illustrated in Figure 4, the routine L_SPX16 envisages that initially a verification pulse will be applied to a number of memory

5   cells equal to the current soft-programming multiplicity (16-bit) (block 400) and that then it will be verified whether there are depleted memory cells or whether, instead, all the memory cells are erased (whether DATA_OK? = 0 or 1, respectively, at block 410).

If all the memory cells are erased (output 1 or YES from block 410),

10   then 16-bit single soft-programming multiplicity is forced, which is also stored in the variable SPX16_MEM (block 420), whereas if there are depleted memory cells (output 0 or NO from block 410), then it is verified whether a pre-set maximum number of attempts has been reached (shown as MAX ATTEMPT? in block 430).

If this maximum number of attempts has not yet been reached

15   (output NO from block 430), then the routine "SP_Pulse" is recalled, whereas if the maximum number of attempts has already been reached (output YES block 430), then control returns to block 420, in which 16-bit single soft-programming multiplicity is forced (shown as SET SPX16 in block 420), and thence block 300, indicated by block B in Figures 3 and 4, of the routine "SP_Pulse" (Figure 3,

20   above) is reached.

From an examination of the characteristics of the method of erasure according to the present invention, the advantages that it affords are evident.

In particular, the present method of erasure enables a soft-programming time approximately 50% shorter to be obtained than what is

25   obtainable using current soft-programming algorithms, and consequently a total erasure time approximately 5% shorter than that in current memory devices, without the need for any over-sizing of the voltage boosters to supply the current necessary in the worst case.

14

Exploiting the property of the statistical distribution of the depleted memory cells, it is hence possible to halve the soft-programming time, given the same area occupied on the silicon.

Furthermore, given that, for other needs, typically nonvolatile

5 memory devices are already provided with a number of program loads twice what is strictly necessary in programming, the implementation of the present invention does not even require the introduction of further program loads.

If, instead, the space on the silicon so enables, by doubling the number of program loads already present on the memory devices, it would be

10 possible to reduce still further the soft-programming time and thus the overall erasure time.

All of the above U.S. patents, U.S. patent application publications, U.S. patent applications, foreign patents, foreign patent applications and non-patent publications referred to in this specification and/or listed in the Application

15 Data Sheet, are incorporated herein by reference, in their entirety.

Finally, it is clear that modifications and variations can be made to the method of erasure described and illustrated herein, without thereby departing from the sphere of protection of the present invention, as defined in the annexed claims.

20 854063.745 / 786341_1.DOC